M3 Milestone Report

1. We have completed the recommendation algorithms for podcasts, movies, and songs which utilize our LSTM model. We also compiled our final datasets which we will pull from to recommend to the user. Our final problem now is figuring out how to integrate our backend code into a website that can take a user input and output our recommendations.

For our music recommendation model, we tried two different methods, the first using a Spotify music dataset created by Yamac Eren Ay on Kaggle, and the second method by creating a dataset by scraping information off a website (Allmusic.com) that categorizes music by moods. We found the first method to be more effective due to the larger number of songs which will ultimately lead to more options for the user. The first method algorithm first attempts to generate a probability distribution across our 6 possible emotions for a user input string. For example, a sentence string such as “I am happy” might have a probability distribution of [sadness = 0, anger = 0, love = 0.2, surprise = 0, fear = 0, happy = 0.8]. This distribution must add to 1. After using this function on the input, we use Pandas to create a data frame for the Kaggle dataset while dropping irrelevant columns. The column categories that are most relevant in the recommendation algorithm are “popularity”, “danceability”, energy”, and “valence”. Details on this can be found on the Spotify Developer API website. A score is generated for every track in the dataset based on these parameters and the user input distribution. Then finally the numpy random choice function is used to “randomly” (using the distribution) choose a Spotify Id which has a high “score”.

For our podcast recommendation model, we cleaned up the dataset to have only relevant genres based on its value count and relevance to our model. Then LSTM was done to classify the input text under five podcast genres. The first layer in our model is the embedded layer that uses 100 length vectors to represent each word. We then used SpatialDropOut1D to perform variational dropout in our model. The next layers are the LSTM layer and the output layer which creates 5 genres as output. We used softmax for activation function as it is a multiclass classification and categorical cross entropy for the loss function. The output genre is then used to recommend its respective podcasts from the dataset. At the end, we were able to get the prediction accuracy of 82% for podcast recommendation.

Finally, we were able to solve the formatting issues that we had with our sentiment analysis model where nearly 5% of sentences became shifted. This problem was caused by whitespace being tokenized which was inconsistent with how the code was meant to be used. Now that all the sentences are correctly formatted, we were able to increase the overall prediction accuracy of our model to 91%.

2. As a group, we decided to focus on creating a website instead of a mobile application. This feature change was made to simplify embedding our models, datasets and recommendation systems using Django. For recommendation after visualizing the dataset, we came to the conclusion that the topic modelling was not useful because the dataset has over 13000 podcasts with its descriptions and this can generate over 25

topics. Because of this, our LSTM model was not able to predict all these classes with good accuracy.

3. The challenge we currently face is how we can integrate our backend code into our front end for our final product. We plan on utilizing the Django Python framework to develop and integrate our application into a website.

4. Max: Used the 600k track Spotify Kaggle dataset created by Yamac Eren Ay and wrote an algorithm to generate a song recommendation for the user based on the user's input. This input was generated by our previously trained LSTM model. The dataset can be found here: https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks/activity

Sijan: Used iTunes podcast dataset created by Roman6335 and wrote an algorithm to generate podcast recommendations for the user based on the user's input. The dataset can be found here:

https://www.kaggle.com/roman6335/13000-itunes-podcasts-april-2018

Kaleb: Used selenium python libraries to scrape information off 'Allmusic.com' which had lists of songs along with their respective information separated into mood categories. With the information from the website, minor adjustments were made including adding new sentiments. Then all of this data was converted into a CSV file for the use of matching the sentiments from our original sentiment analysis model.

Ryan: Parsed title.basics and title.ratings from the IMDb data set of movies https://www.imdb.com/interfaces/ to create a single file containing the title, synopsis, genre and rating of each of the listed films. Spent time learning Django to build the web app.